Collaborative Modelling and Co-simulation

Tools and Techniques for Designing Embedded Systems

John Fitzgerald
Fuyuki Ishikawa
Peter Gorm Larsen

Newcastle University
AARHUS UNIVERSITY

Programme

10:00 – 11:30: Introduction to DESTECS/Crescendo, Co-modelling and Co-simulation

11:30 – 12:30: Quick Introduction to VDM

12:30 – 13:30: Lunch

13:30 – 15:30: Practical Trial with Line-following Robot

15:30 – 16:00: Coffee Break

16:00 – 17:30: Application Experiences, Implications, Discussions

The DESTECS Team

DESTECS

Design Support and Tools for Embedded Control Systems

(Jan 2010 – Dec 2012), www.destecs.org

John Fitzgerald, Kenneth Pierce, Carl Gamble, Claire Ingram, Peter Gorm Larsen, Kenneth Lausdahl, Augusto Ribeiro, Joey Coleman, Kim Bjerge, Sune Wolff, José Antonio Esparza Isasa, Claus Ballegard Nielsen, Martin Peter Christensen, Jan Broenink, Xiaochen Zhang, Yunyun Ni, Angelika Mader, Jelena Marinčić, Christian Kleijn, Peter Visser, Frank Groen, Marcel Groothuis, Peter van Eijk, Dusko Jovanovic, Jan Remijnse, Eelke Visser, Michiel de Paepe, Koenraad Rombaut, Yoni de Witte, Roeland van Lembergen, Wouter Vleugels, Bert Bos, Jeffrey Simons

Introduction

John Fitzgerald
Peter Gorm Larsen

Newcastle University
AARHUS UNIVERSITY

Background

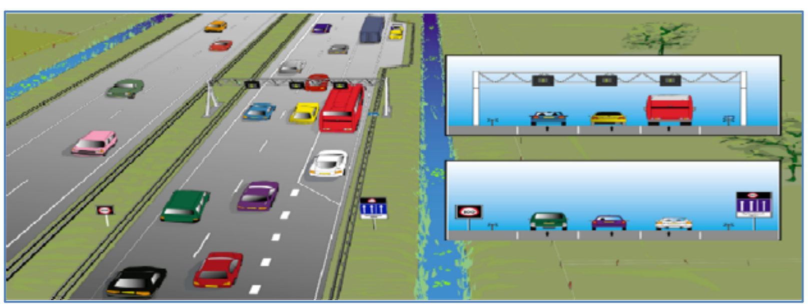
- Computers become smaller, more capable, ubiquitous
- ~6.8 billion mobile phone accounts
- Look at automobiles:
  - 80 processors & >100M LOC in a high-end vehicle
  - Recall costs immense
    - Brand loyalty: 55% → 39% if you experience 3+ problems

80s

90s

00s

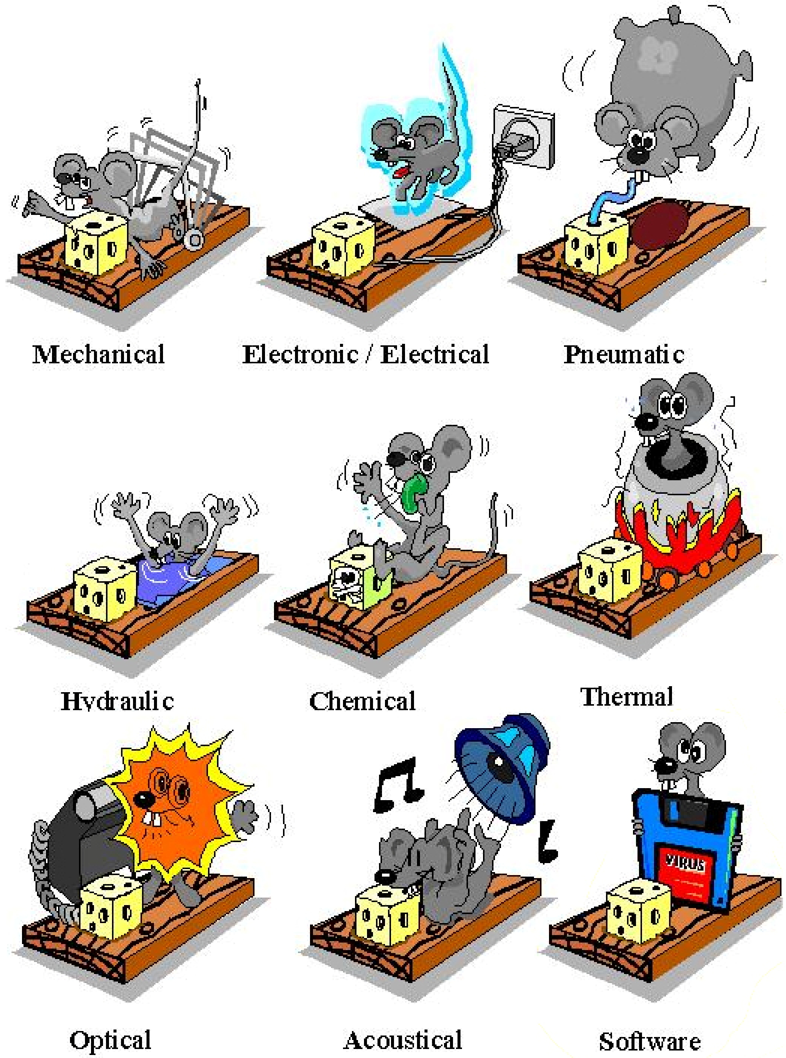
10s



Background

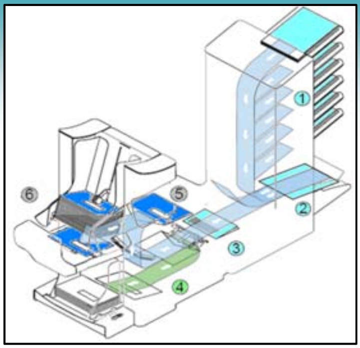

We offer three kinds of service:
GOOD - CHEAP - FAST
You can pick any two
GOOD service CHEAP won't be FAST
GOOD service FAST won't be CHEAP
FAST service CHEAP won't be GOOD

Background



- Problem decomposition into disciplines
- Concurrent engineering required to improve time to market
- ... but important properties are multidisciplinary
- ... and so weaknesses are exposed late (integration)
- So: how to cross the boundaries between disciplines?

Background: Co-modelling



Software:
- Discrete
- Complex logic

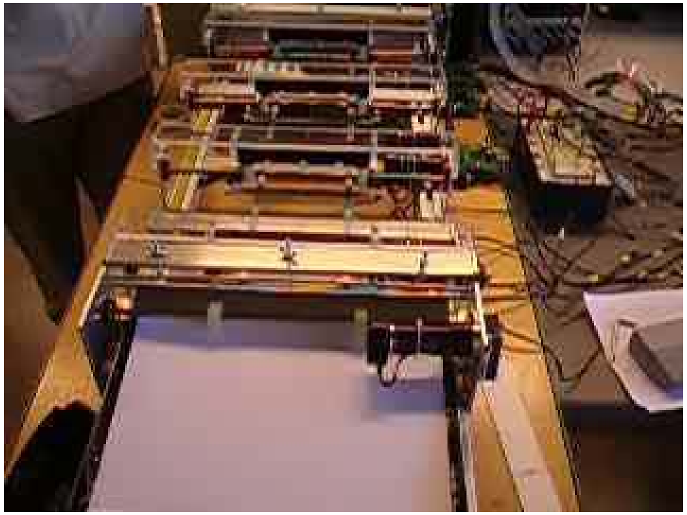
Mind the Gap!

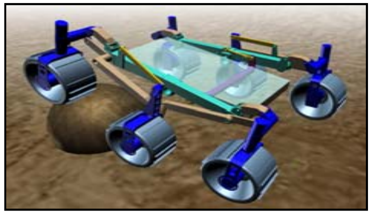
Physics:
- Continuous
- Numerical

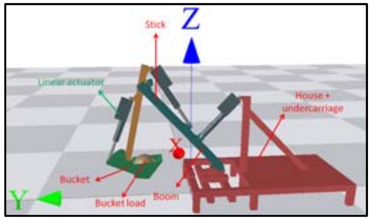
Co-model

DE Model — **CT Model**

Co-model Interface



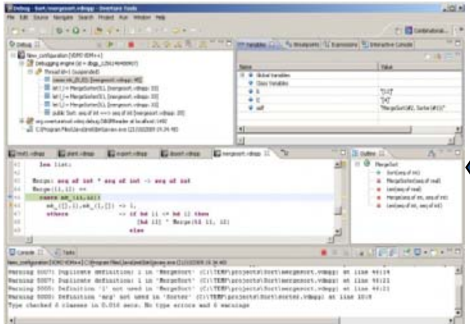


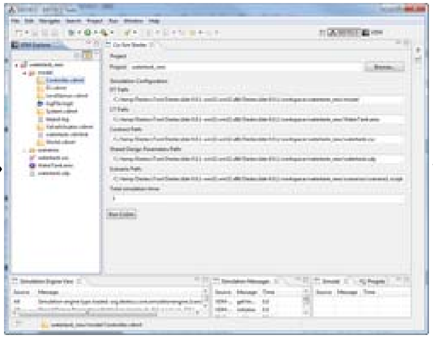


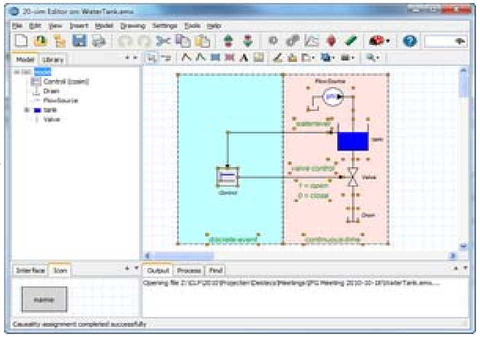
Background: Co-simulation

Discrete-Event Simulator ⟷ **Co-Simulation Engine** ⟷ **Continuous-Time Solver**

Overture | **Crescendo** | **20-sim**



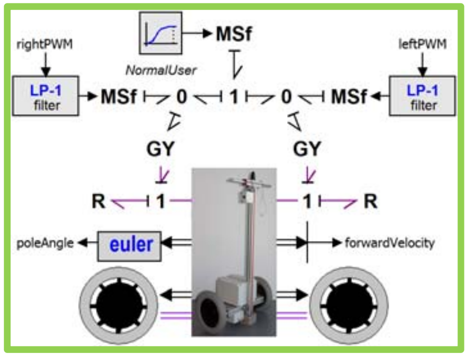




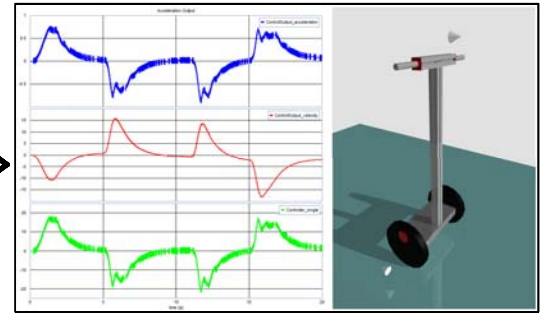
Co-model Support

```
class Controller

instance variables
  -- sensors
  private angle: real;
  private velocity: real;
  -- actuators
  private acc_out: real;
  private vel_out: real;
  -- PID controllers
  private pid1: PID;
  private pid2: PID;

operations
  public Step : () ==> ()
  Step() == duration(20) (
    dcl err: real := velocity - angle;
    vel_out.Write(pid2.Out(err));
    acc_out.Write(pid1.Out(angle));
  );

  public GoSafe : () ==> ()
  GoSafe() == (
    vel_out.Write(0);
    acc_out.Write(0);
  );

thread
  periodic(1E6,0,0,0)(Step); -- 1kHz

end Controller
```

+

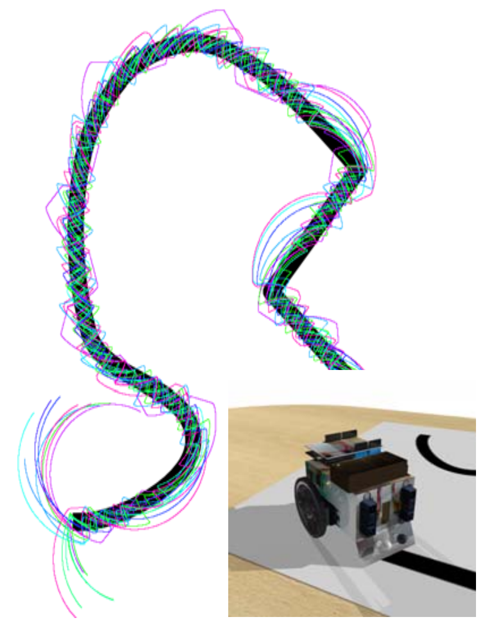


⇒



⇒

- Products: tools (Crescendo) method guidelines (notably fault modelling)
- Automated Co-model Analysis (sweeps, ranking)
- Reduced design iteration/cost in transport, machine design, high-speed paper processing and baggage handling!



DESTECS: Design Support and Tools for Embedded Control Systems

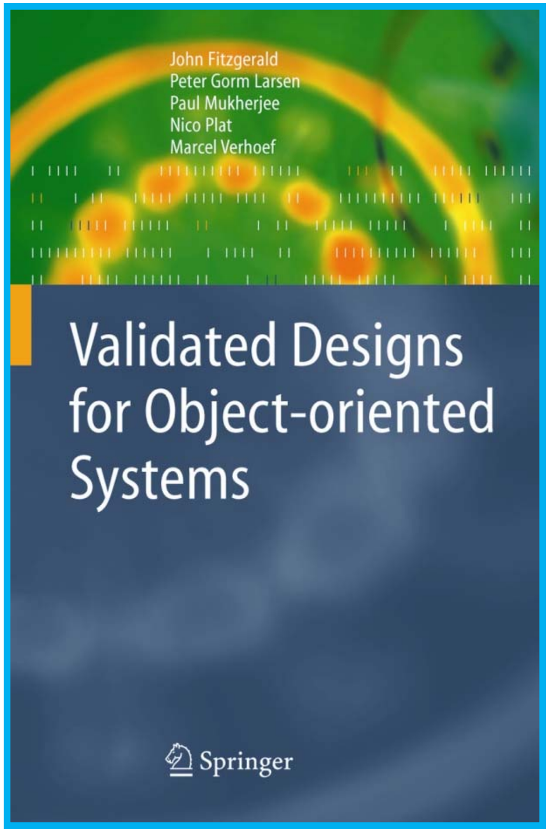
EU FP7 INFSO-ICT-248134 (Jan 2010 – Dec 2012)

Océ
Airbus
Nokia
Siemens
Martin Group
Atlas Copco

Dutch Space
ESA
FKI Logistex
Darwind
ASML
Assembleon

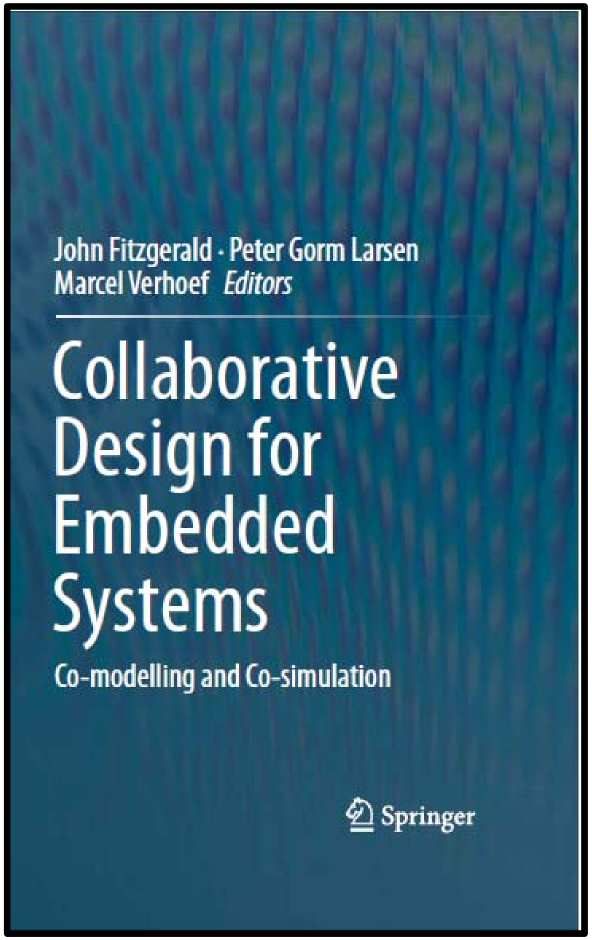
Vestas
Grundfos
Volvo
Bang & Olufsen
MBDA
Terma

Reference Books



Baseline Discrete Event Modelling

Baseline Continuous Time Modelling



Co-Modelling

Co-modelling and Co-simulation

John Fitzgerald
Peter Gorm Larsen

Newcastle University
AARHUS UNIVERSITY

Model-driven Design

- Modern systems are complex
- To cope with this, we can build models beforehand
  - To perform analysis (e.g. static analysis, proof, model checking, **simulation**)
  - Clarify our assumptions
  - Evaluate potential designs
  - Avoid expensive prototypes
- Different modelling paradigms for different aspects

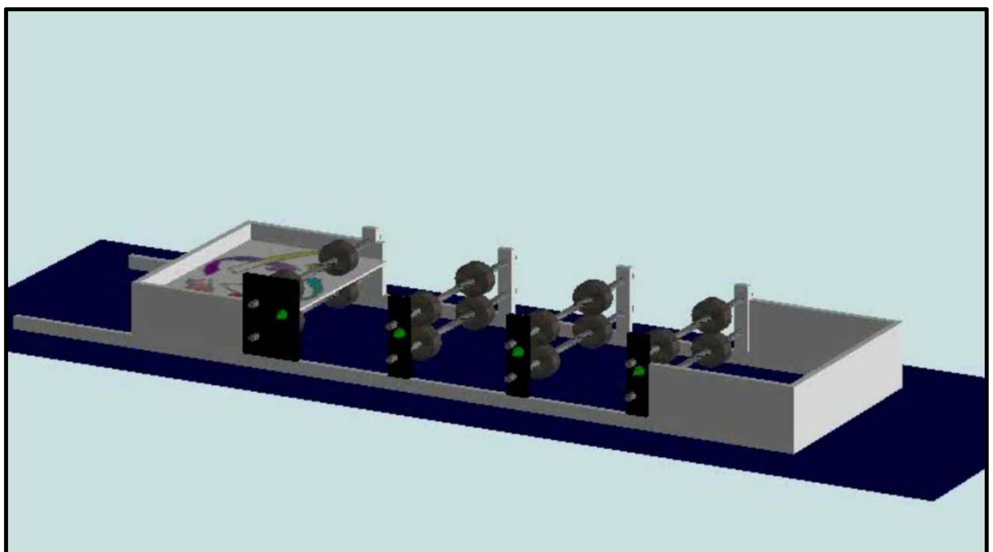
Modelling of Software and Physics

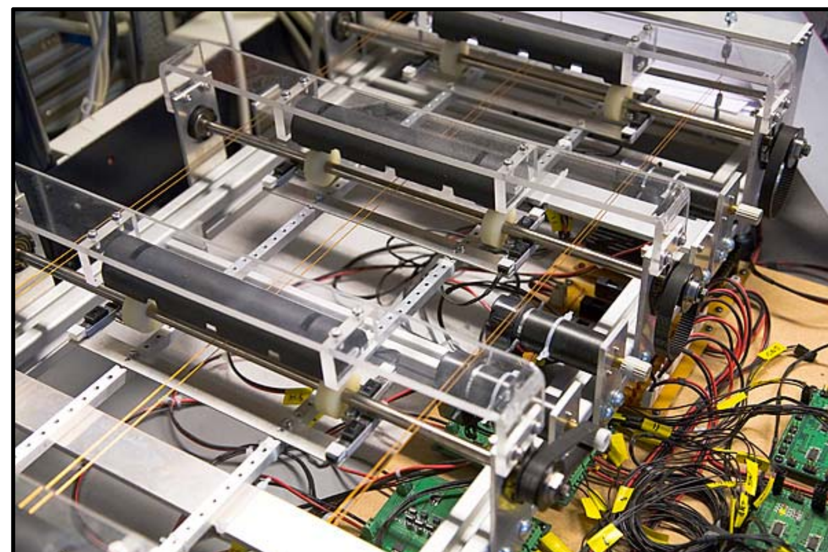
- Typically **discrete-event (DE),** e.g. VDM-RT
- In simulation, only the points in time at which the state changes are represented
- Good abstractions for software,
  - e.g. data types, object-orientation, threading
- Less suited for physical system modelling

- Typically **continuous-time (CT),** e.g. differential equations
- In simulation, the state changes continuously through time
- Abstractions for disciplines,
  - e.g. mechanical, electrical, hydraulic
- Poor software modelling support
  - only basic programming support; no functions or objects

Embedded Systems

- Interacting computing, physical, human elements
- Increasingly complex logic (e.g. moding) ~80% of control software
- Error detection and recovery
- Collaborative development
- Diverse disciplines cultures, abstractions, formalisms
- Typically tackled separately
- Need for **design space exploration**

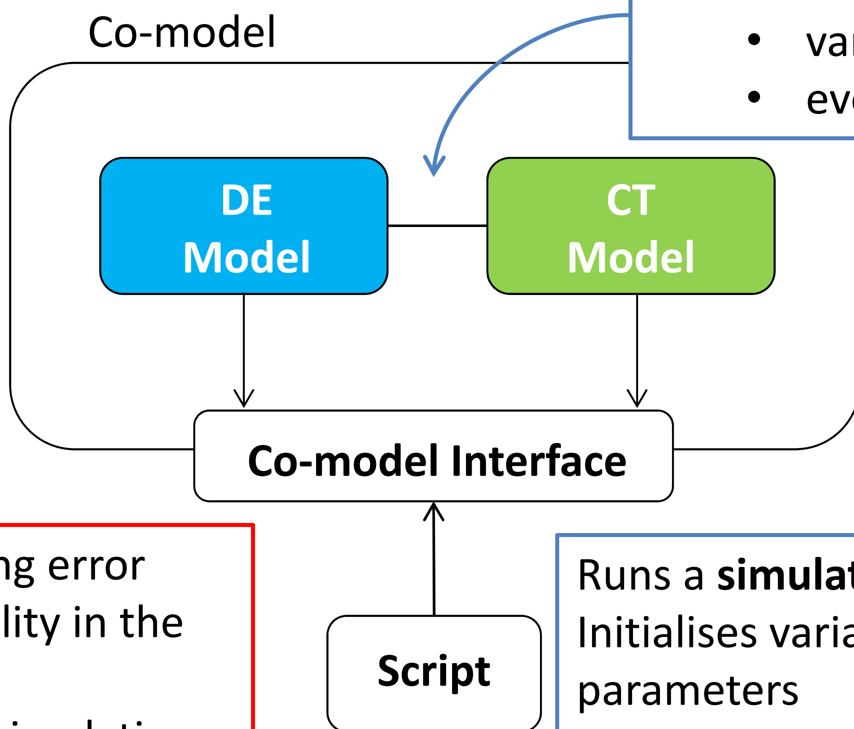




Co-modelling Concepts

Variables modified during run
Design parameters fixed per run

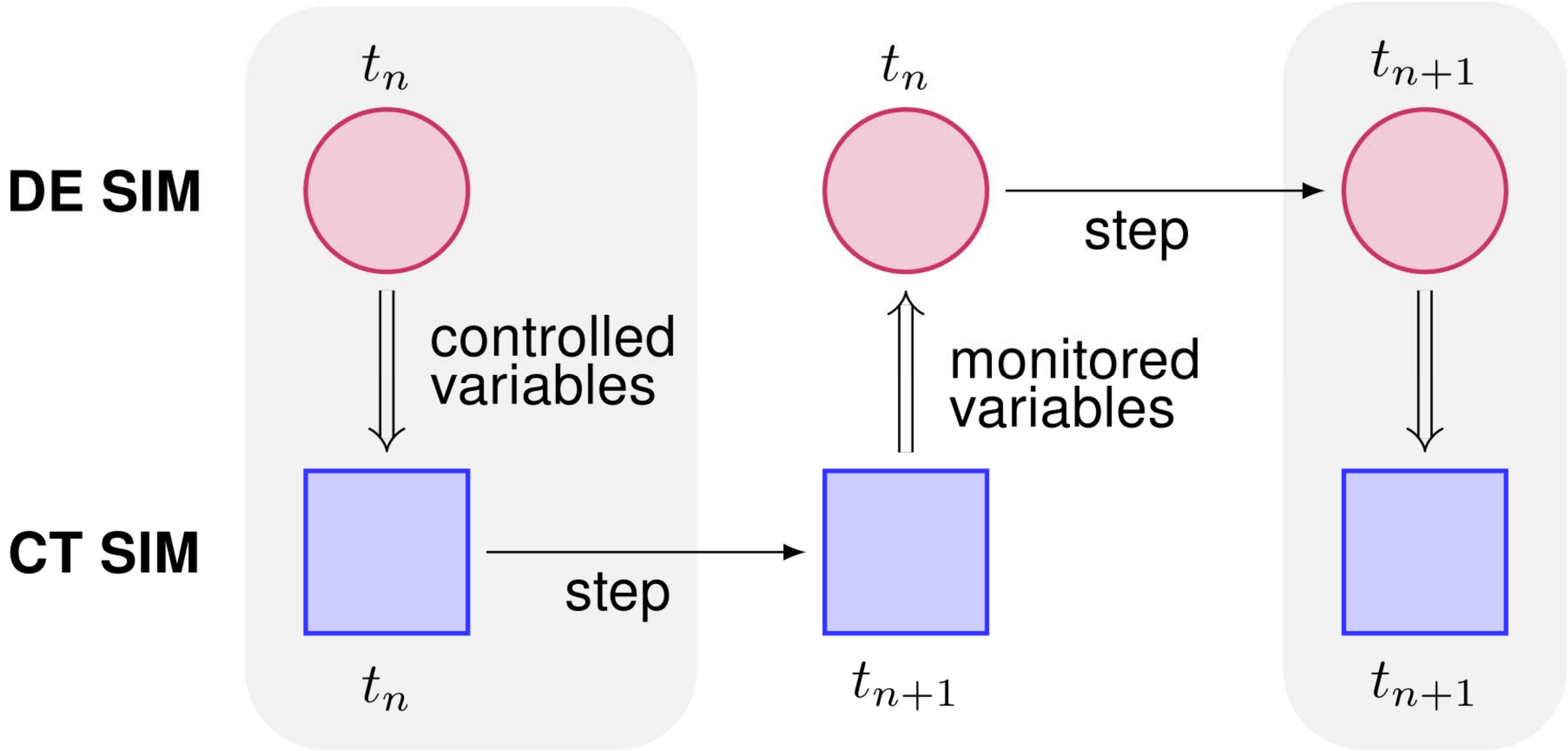
"Contract": shared
- design parameters
- variables
- events



Fault Modelling: including error states & faulty functionality in the model
Faults invoked during a simulation managed by script

Runs a **simulation**
Initialises variables and design parameters
Forces selections and external updates, e.g. set point

Co-simulation Semantics



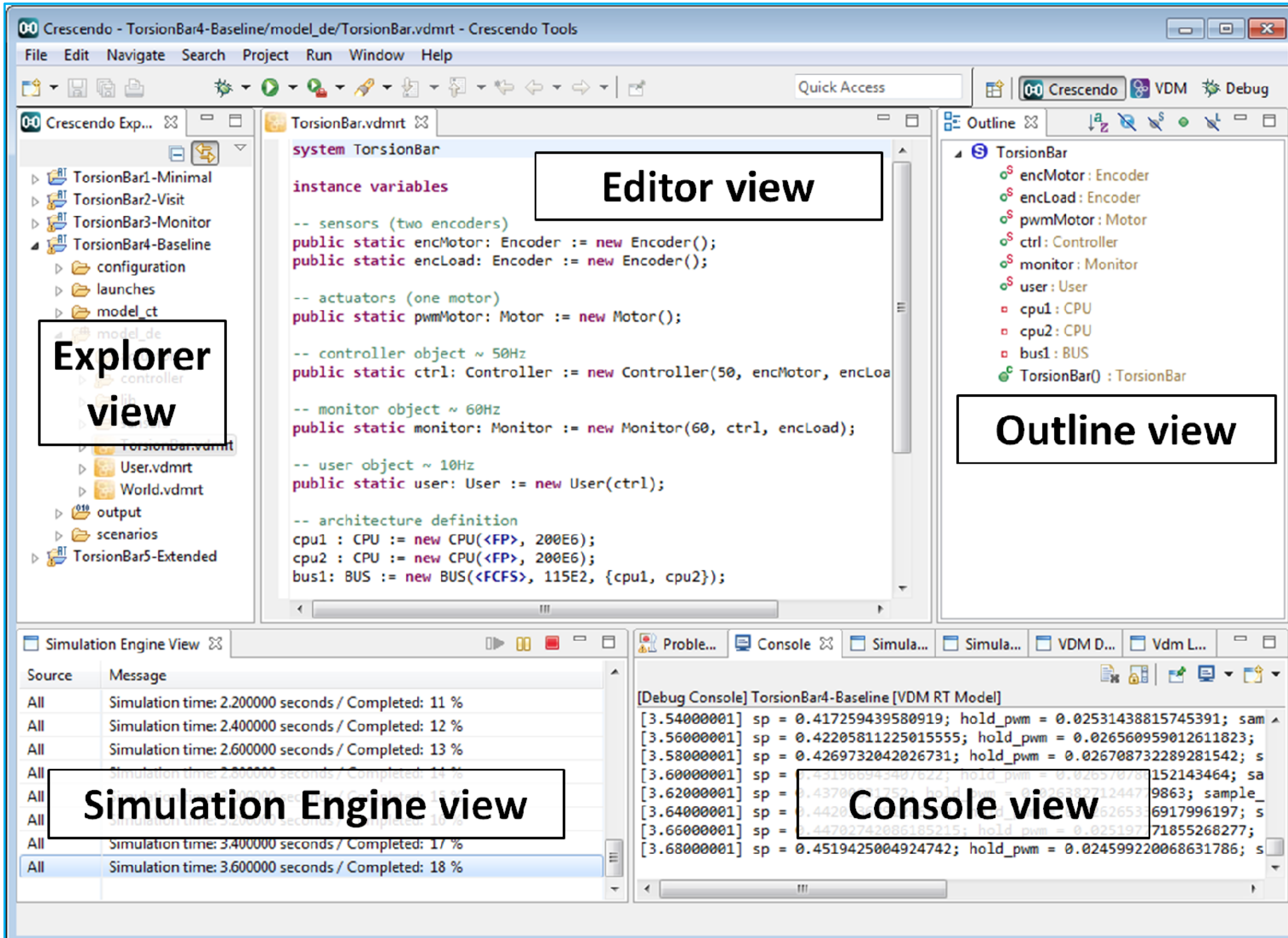
Co-simulation Semantics

- Simulators maintain local state / internal simulation time.
- Co-simulation engine synchronises:
  - shared variables, events, time
- Common time, t_n, at the start of a co-simulation step.
- DE simulator determines step length (to avoid roll-back).
- At t_n, the DE simulator:
  - sets controlled variables
  - proposes duration to for CT simulator to advance (if possible).
- Co-simulation engine tells the CT simulator to advance.

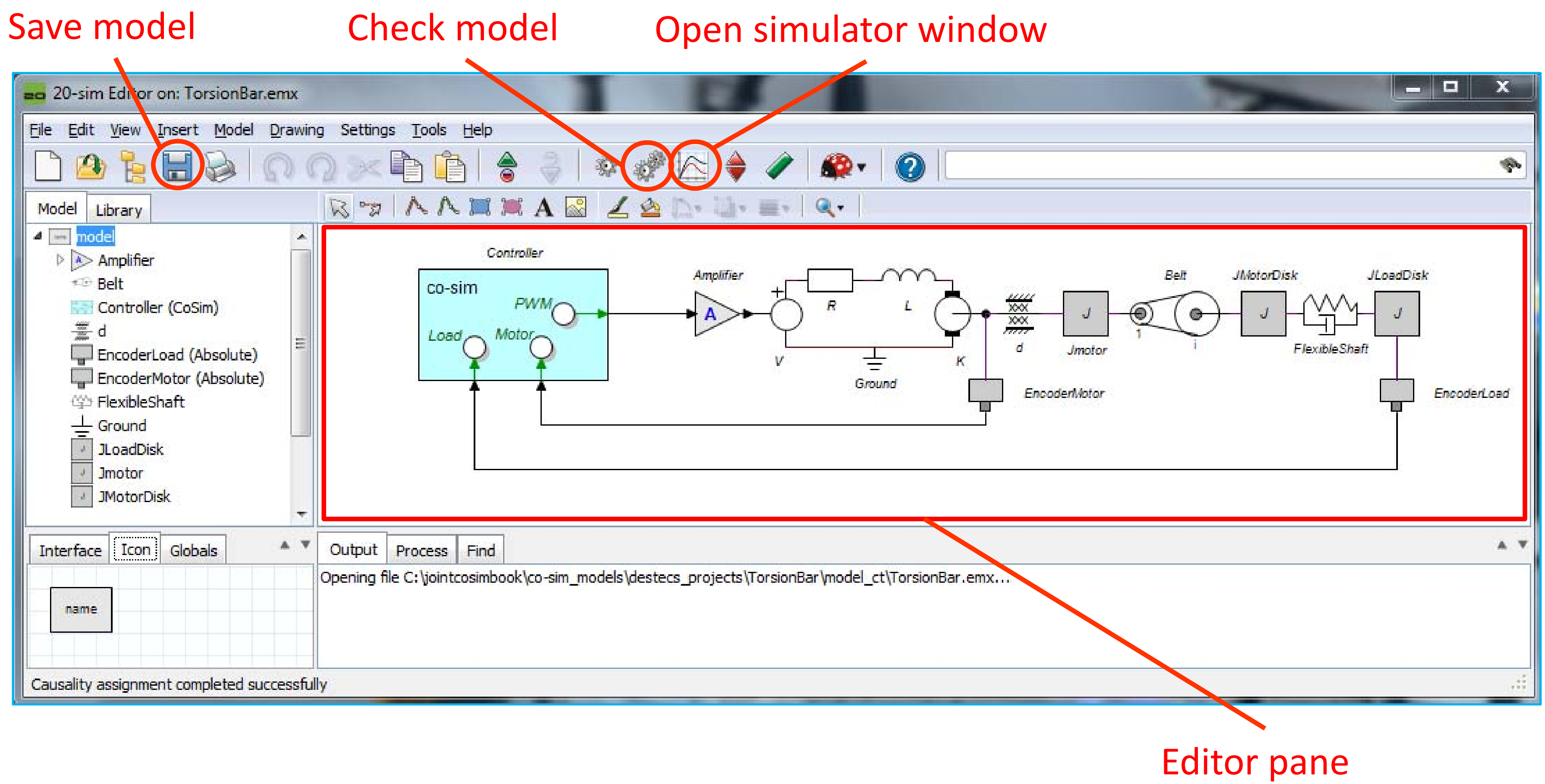
Co-simulation Semantics

- The CT simulator advances. If an event occurs before the proposed step time is reached, CT simulator stops early.
- Once the CT simulator has paused (reaching internal time t_{n+1}), the monitored variables and the actual time reached in the CT simulation are communicated back to the DE simulator.
- The DE simulation then advances so that both DE and CT are again synchronised at the same simulation time.
- *Cycle repeats.*

Crescendo Screenshot



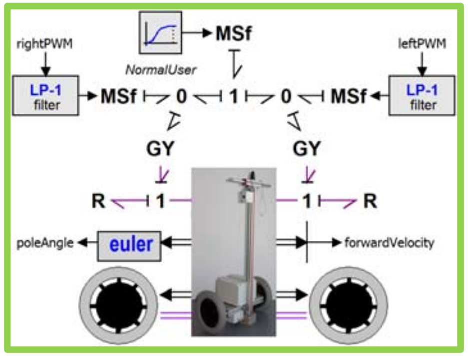
20-sim Screenshot



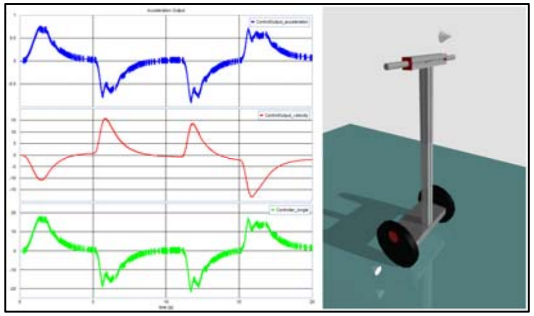
Example: Self-balancing Scooter

```
class Controller

instance variables
  -- sensors
  private angle: real;
  private velocity: real;
  -- actuators
  private acc_out: real;
  private vel_out: real;
  -- PID controllers
  private pid1: PID;
  private pid2: PID;

operations
  public Step : () ==> ()
  Step() == duration(20) (
    dcl err: real := velocity - angle;
    vel_out.Write(pid2.Out(err));
    acc_out.Write(pid1.Out(angle));
  );

  public GoSafe : () ==> ()
  GoSafe() == (
    vel_out.Write(0);
    acc_out.Write(0);
  );

thread
  periodic(1E6,0,0,0)(Step); -- 1kHz

end Controller
```

+

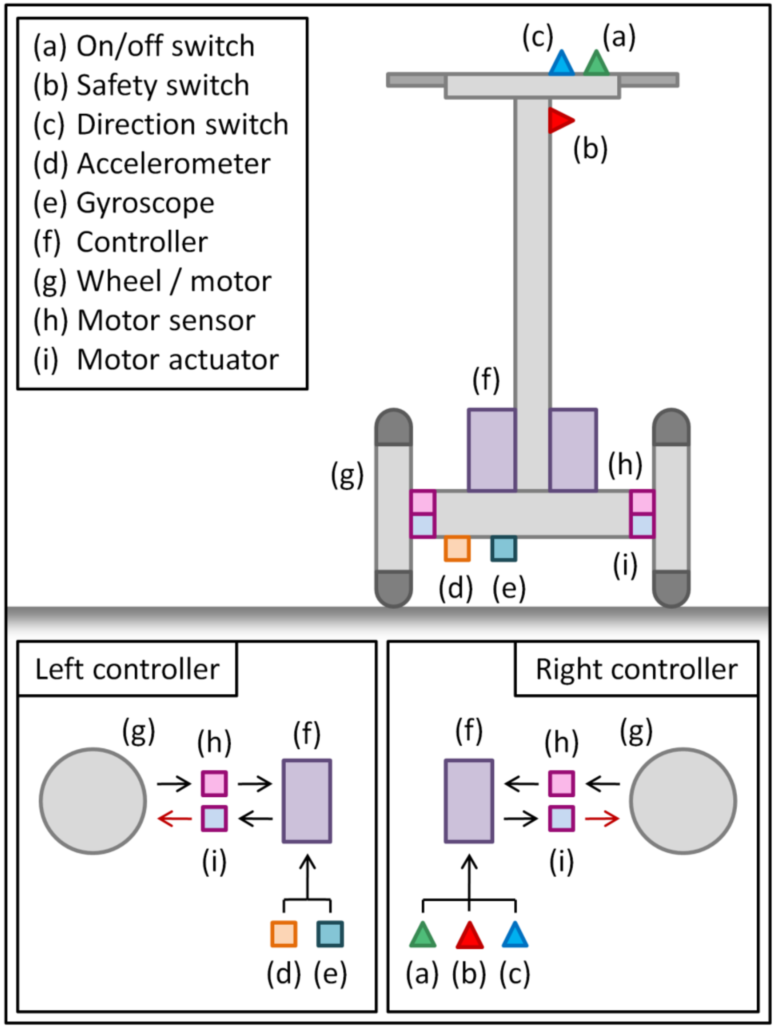


⇒

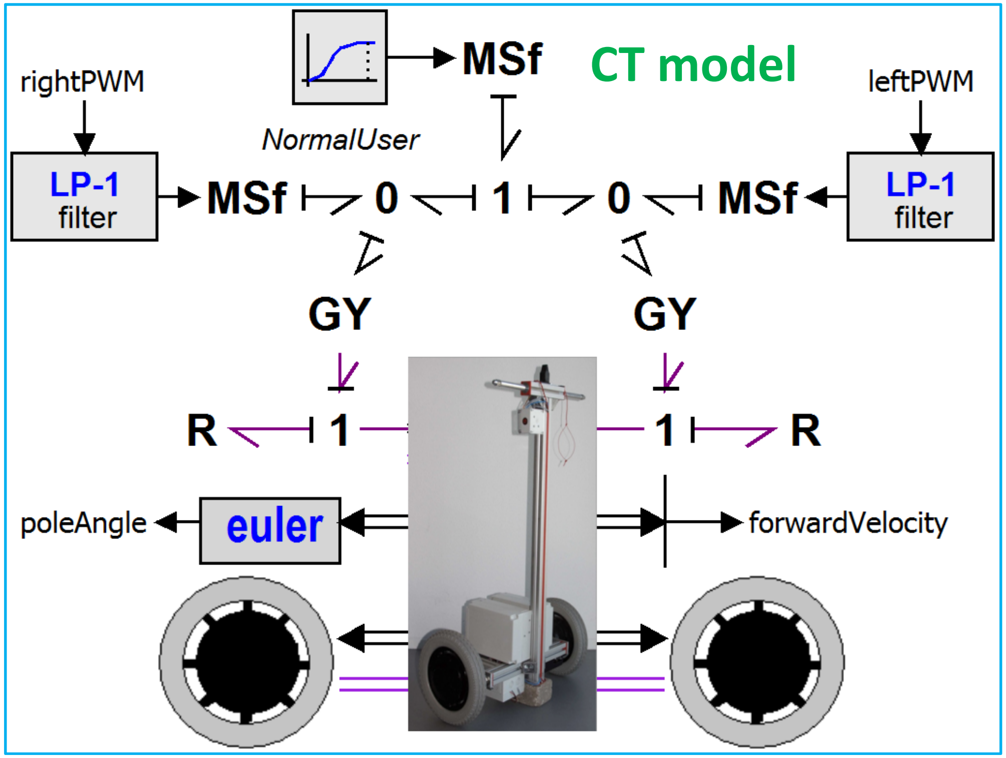


⇒

Example: Self-balancing Scooter



Example: Self-balancing Scooter



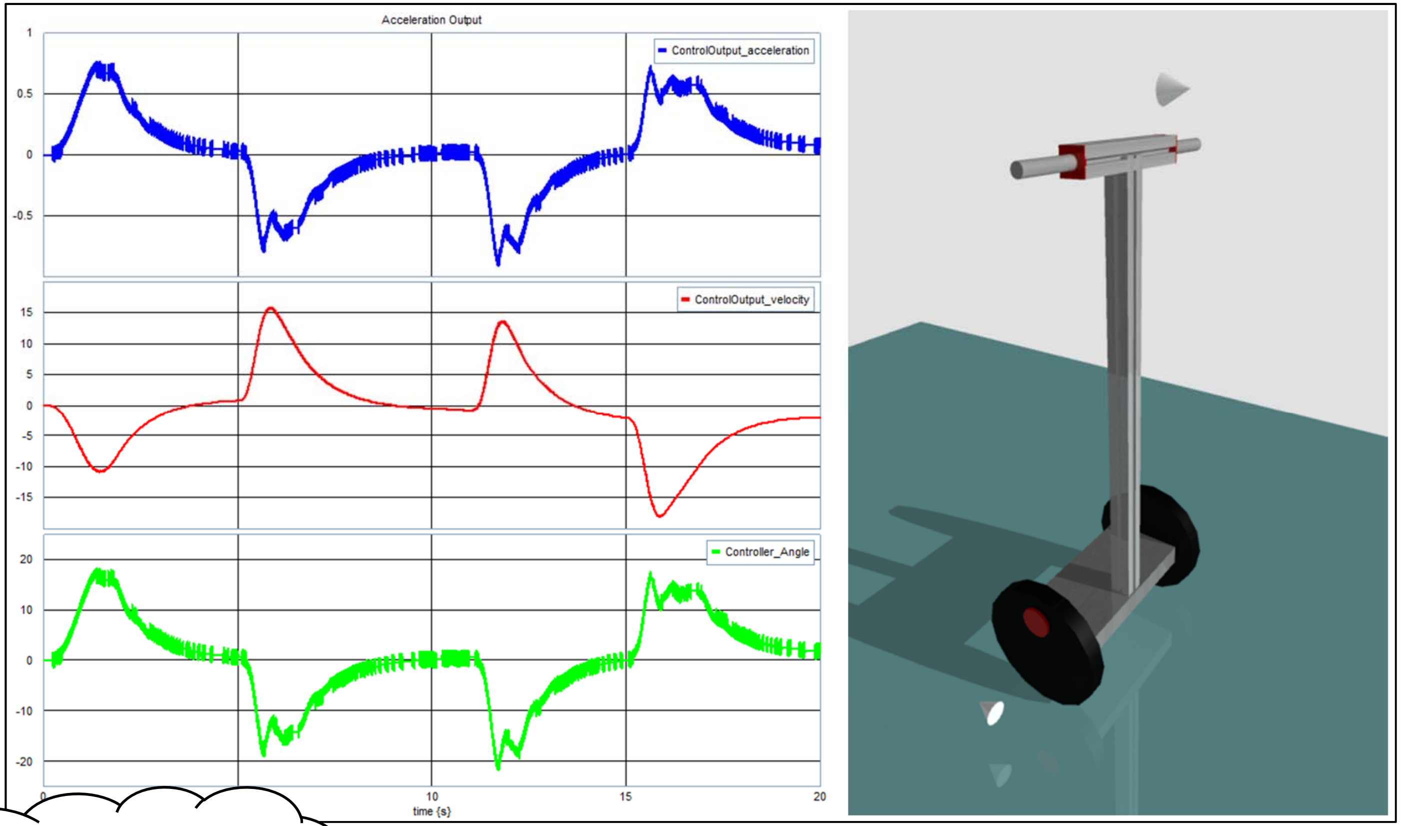
	Name	Type	Notes
controlled	leftPWM	real	range: [-1,1]
	rightPWM	real	range: [-1,1]
monitored	poleAngle	real	range: $[0,2\pi]$
	forwardVelocity	real	

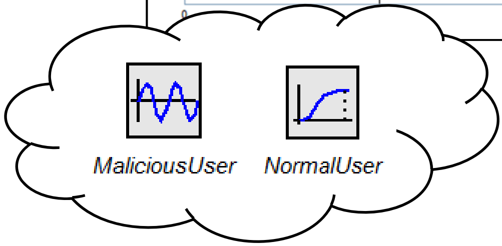
Contract

DE model

```
class Controller

instance variables
  -- sensors
  private angle: real;
  private velocity: real;
  -- actuators
  private acc_out: real;
  private vel_out: real;
  -- PID controllers
  private pid1: PID;
  private pid2: PID;

operations
  public Step : () ==> ()
  Step() == duration(20) (
    dcl err: real := velocity - angle;
    vel_out.Write(pid2.Out(err));
    acc_out.Write(pid1.Out(angle));
  );

  public GoSafe : () ==> ()
  GoSafe() == (
    vel_out.Write(0);
    acc_out.Write(0);
  );

thread
  periodic(1E6,0,0,0)(Step); -- 1kHz

end Controller
```

Example: Self-balancing Scooter





http://www.youtube.com/watch?v=pmLLGYn9Fo8

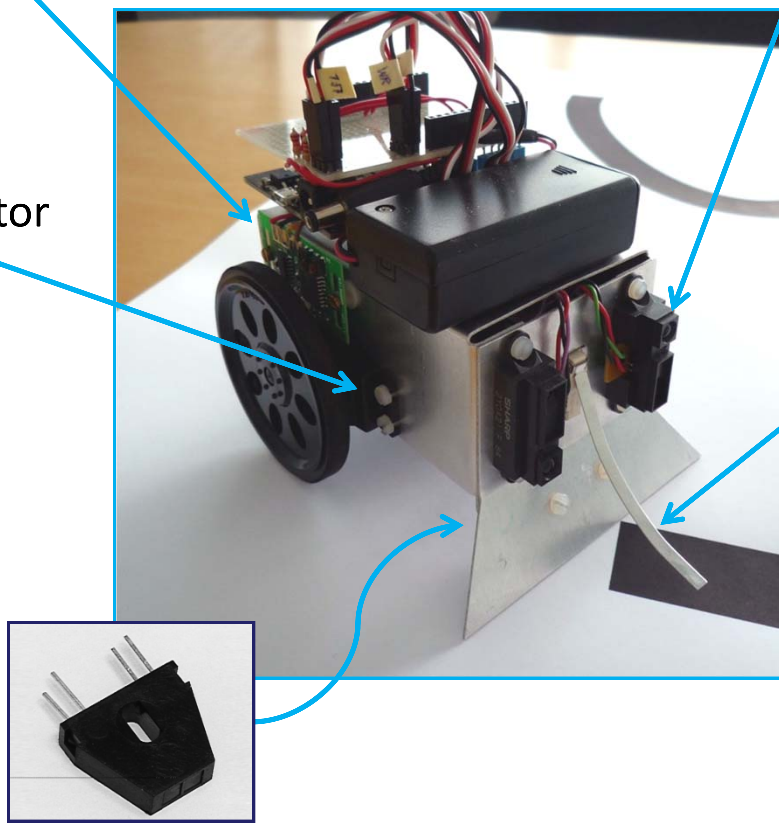
Example: Line-following Robot

wheel encoder

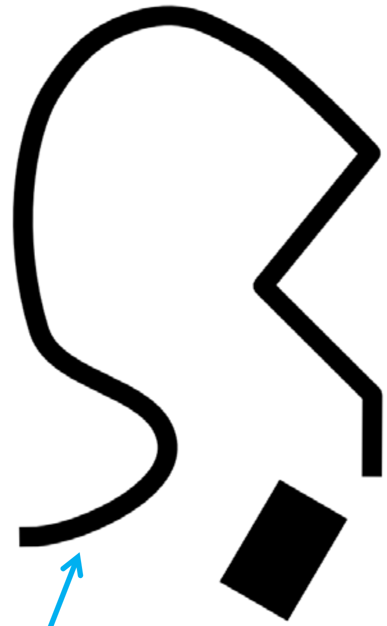
IR distance sensors

servo motor

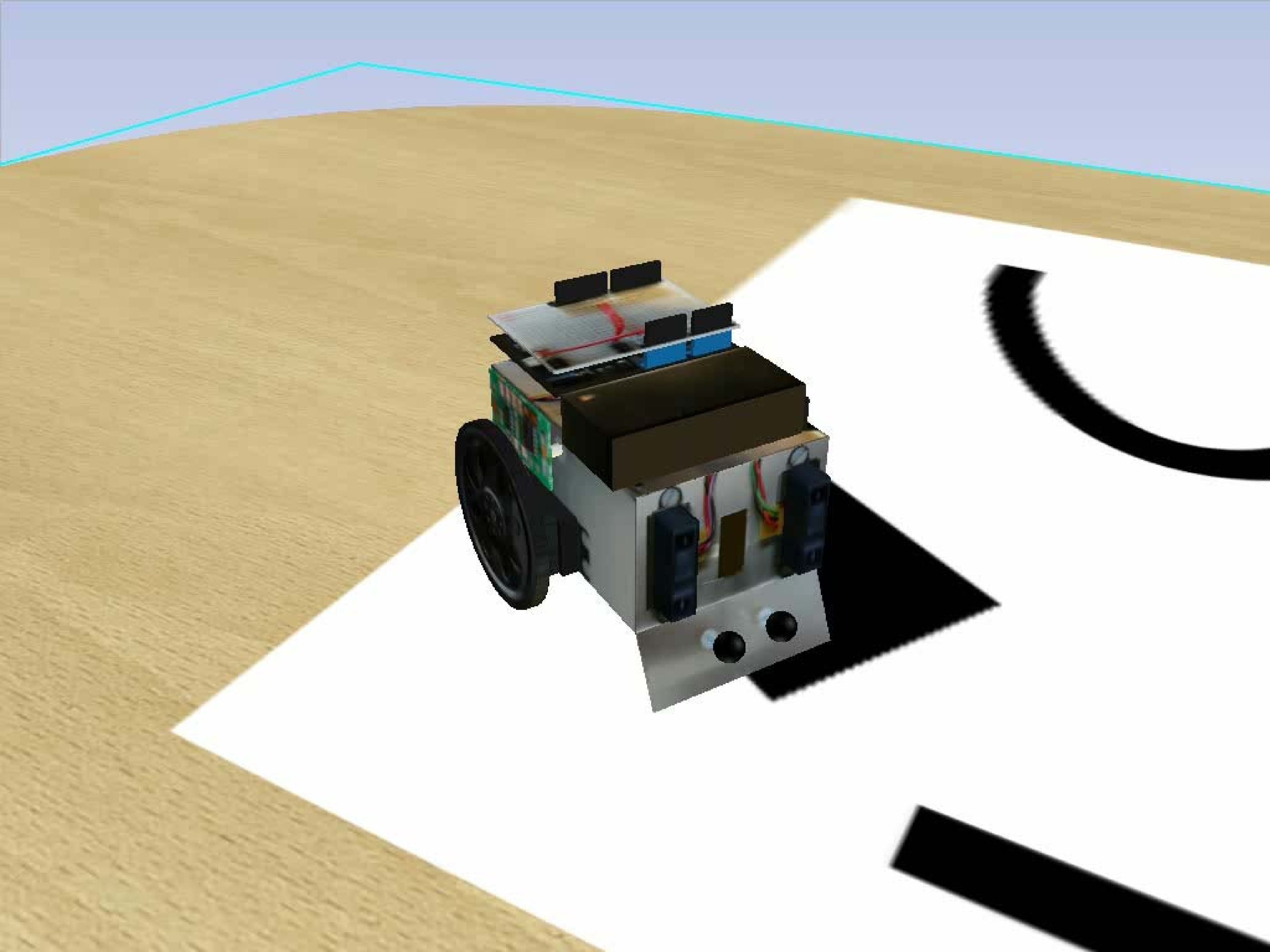
contact switch



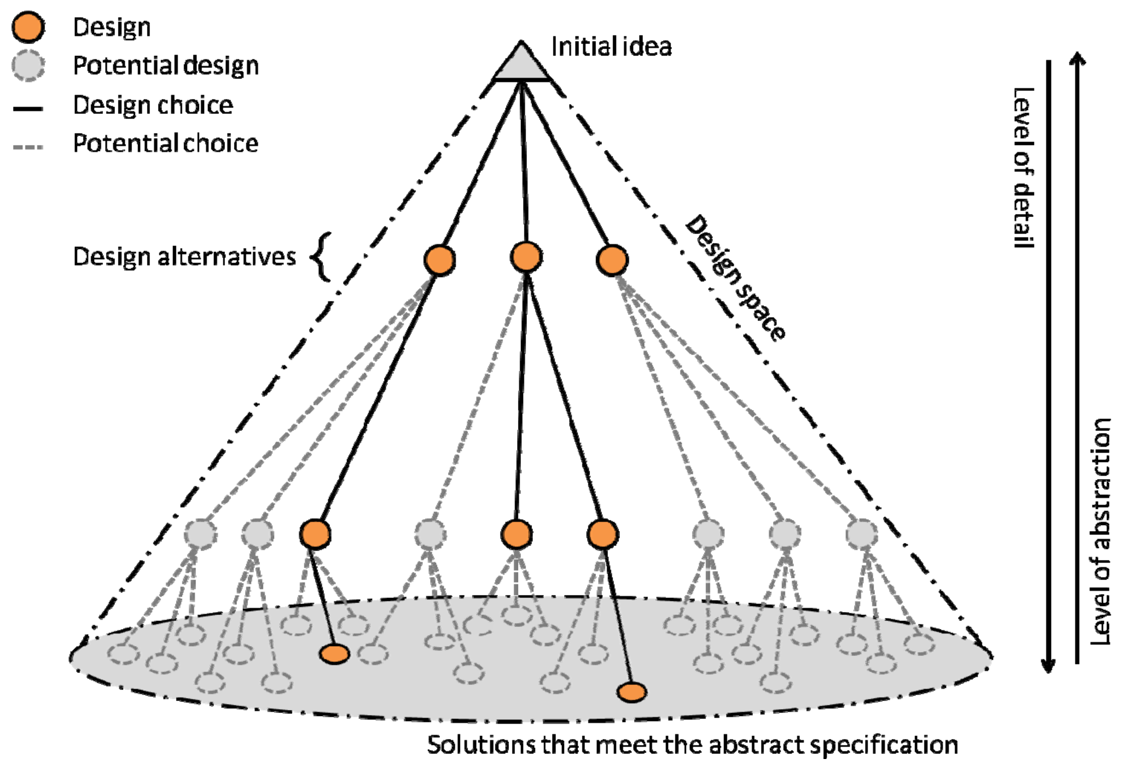
IR line-follow sensors



example path



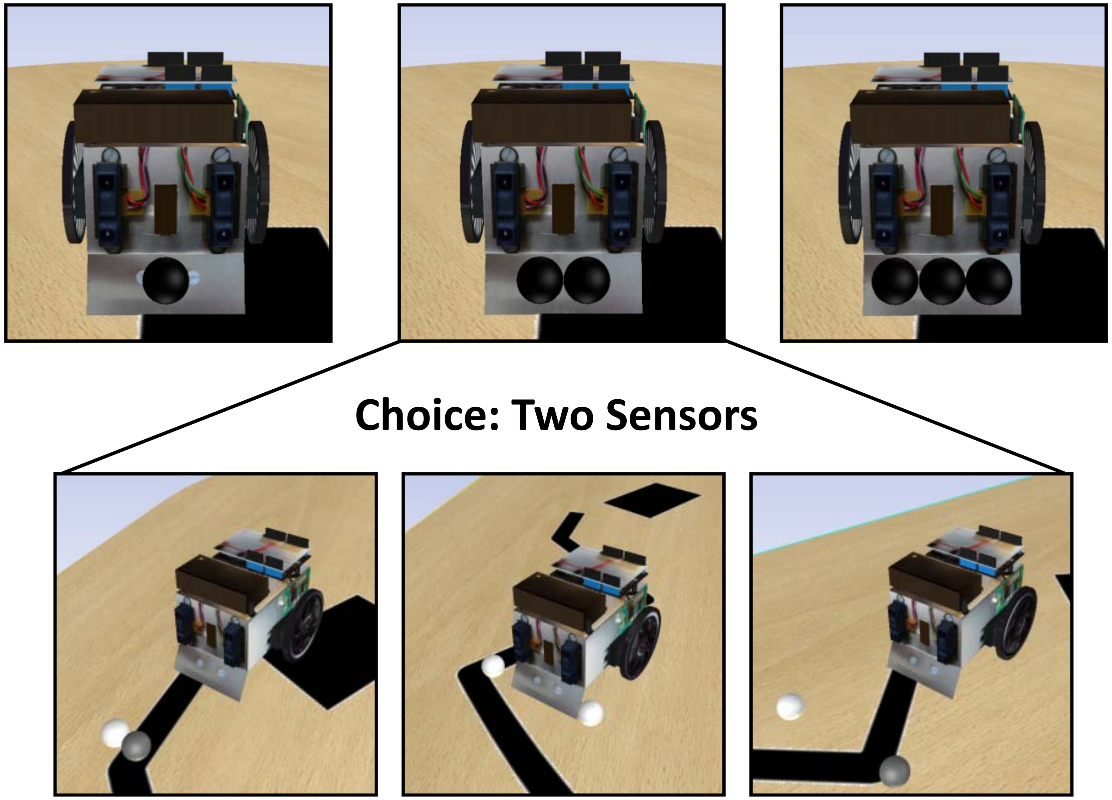
Design-Space Exploration (DSE)



- Selecting alternative designs (based on e.g. cost, performance).
- The alternative selected at each point constrains the range of designs that may be viable next steps.

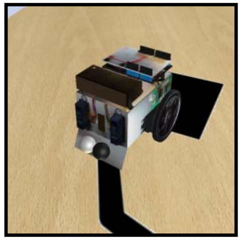
Line-following Robot DSE

- Design choices restrict the design space
- Exploration is making decisions

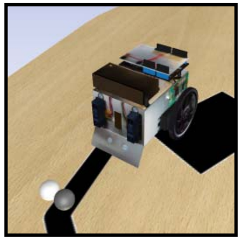


Line-following Robot DSE

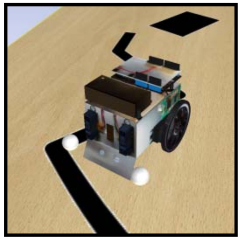
		Longitudinal sensor offset		
		0.01m	**0.07m**	**0.13m**
Lateral sensor offset	**0.01m**	(a)	(b)	(c)
	0.03m	(d)	(e)	(f)
	0.05m	(g)	(h)	(i)



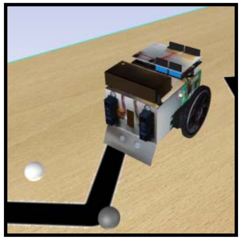
(b)



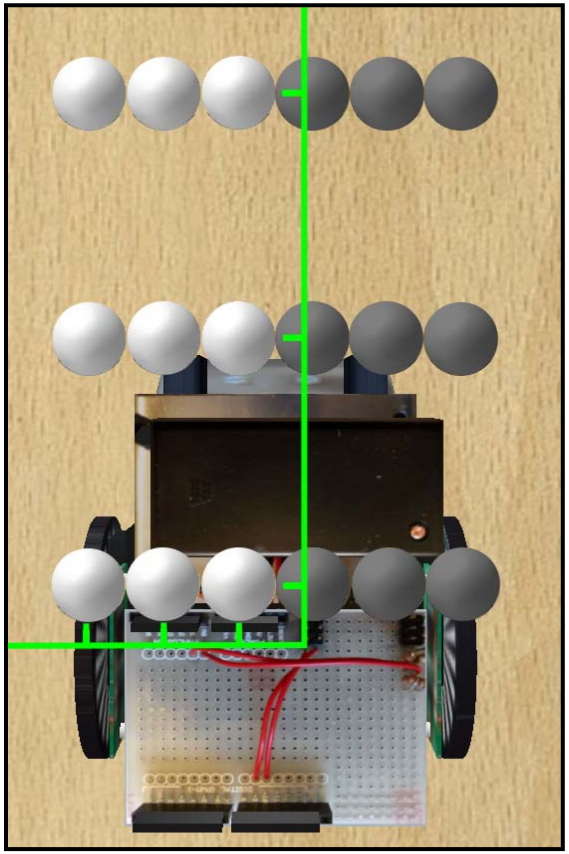
(c)



(h)



(i)

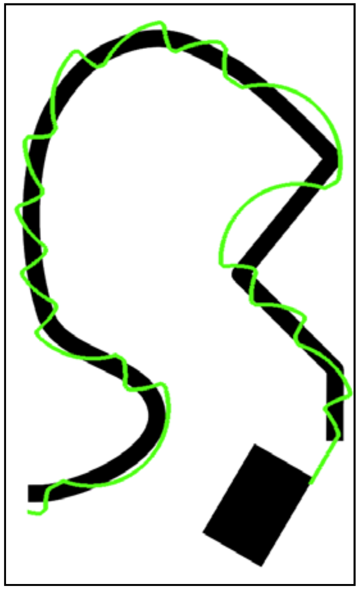


- Two parameters, 9 choices / simulations

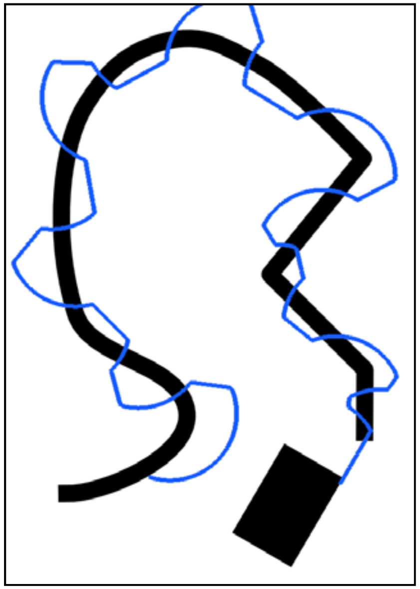
Line-following Robot DSE

- Results can be graphical or numerical
- Designs can be evaluated by ranking functions

		Metric*				
Rank	Design	A	B	C	D	Mean Rank
1	**(b)**	**1**	**5**	**1**	**2**	**2.2**
2	(f)	7	2	4	1	3.5
3	(a)	2	8	2	4	4.0
4	(e)	3	6	3	5	4.2
5	(i)	9	1	5	3	4.5
6	(c)	5	3	6	8	5.5
7	(d)	6	4	7	7	6.0
8	(h)	4	7	8	9	7.0
9	(j)	8	9	9	6	8.0



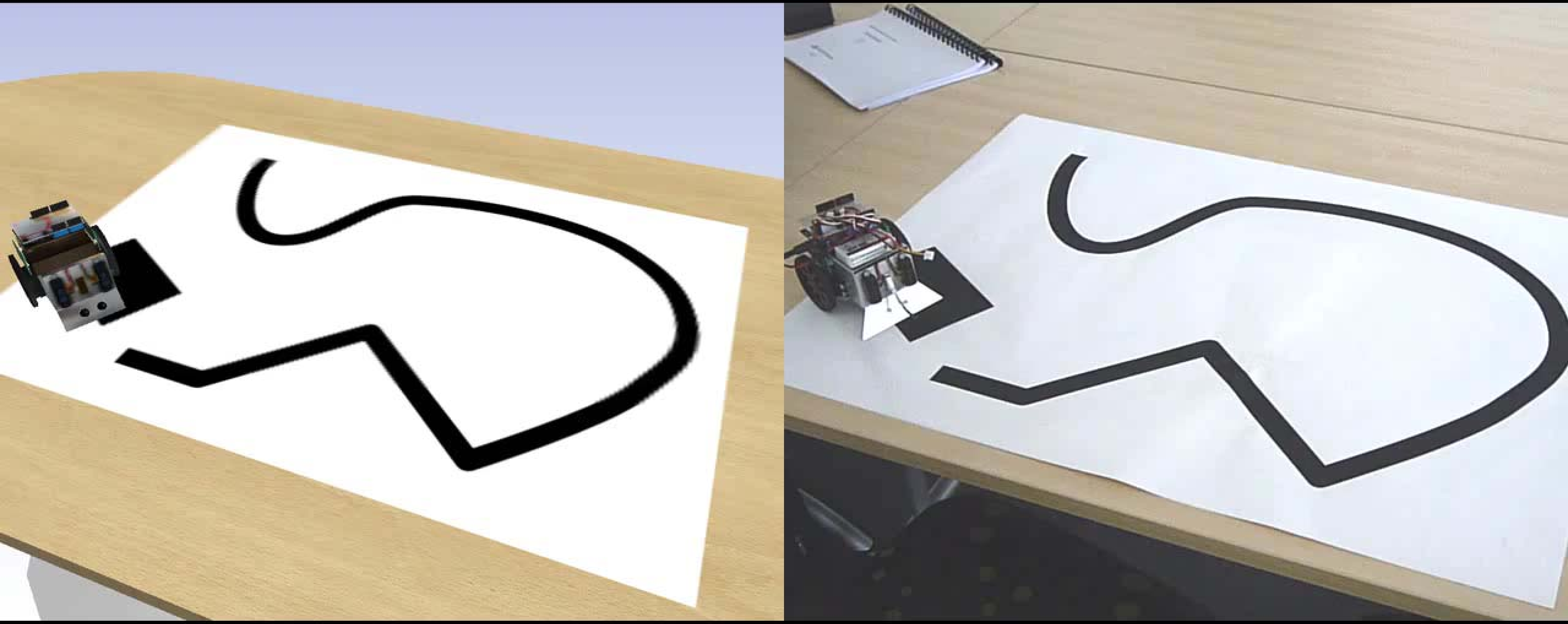
(b)



(h)

*** A = distance, B = energy, C = deviation area, D = maximum deviation**

Line-following Robot

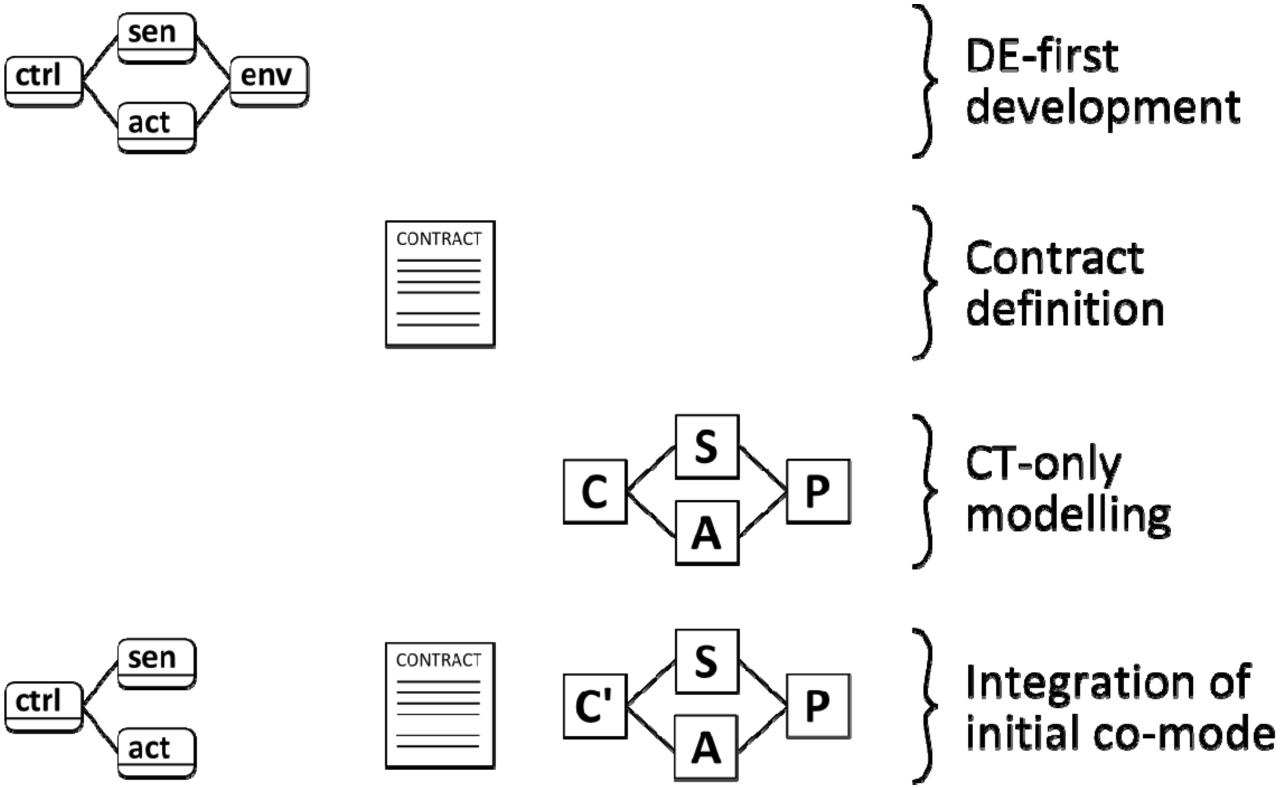


Paths to Initial Co-models

- **DE-first**
  - initial models are produced in the discrete-event formalism; CT model added later. Focus on DE controller first.
- **CT-first**
  - Initial models are produced in the CT tool, with a DE model being introduced later to form a co-model. Focus on modelling the dynamics of the plant.
- **Contract-first**
  - Contract defined, acts as a guide. DE- and CT-models are developed separately but concurrently (DE-first and CT-first, as above). Allows for early testing of constituent models without reliance on a competent counterpart model. The constituent models are then integrated into a co-model.

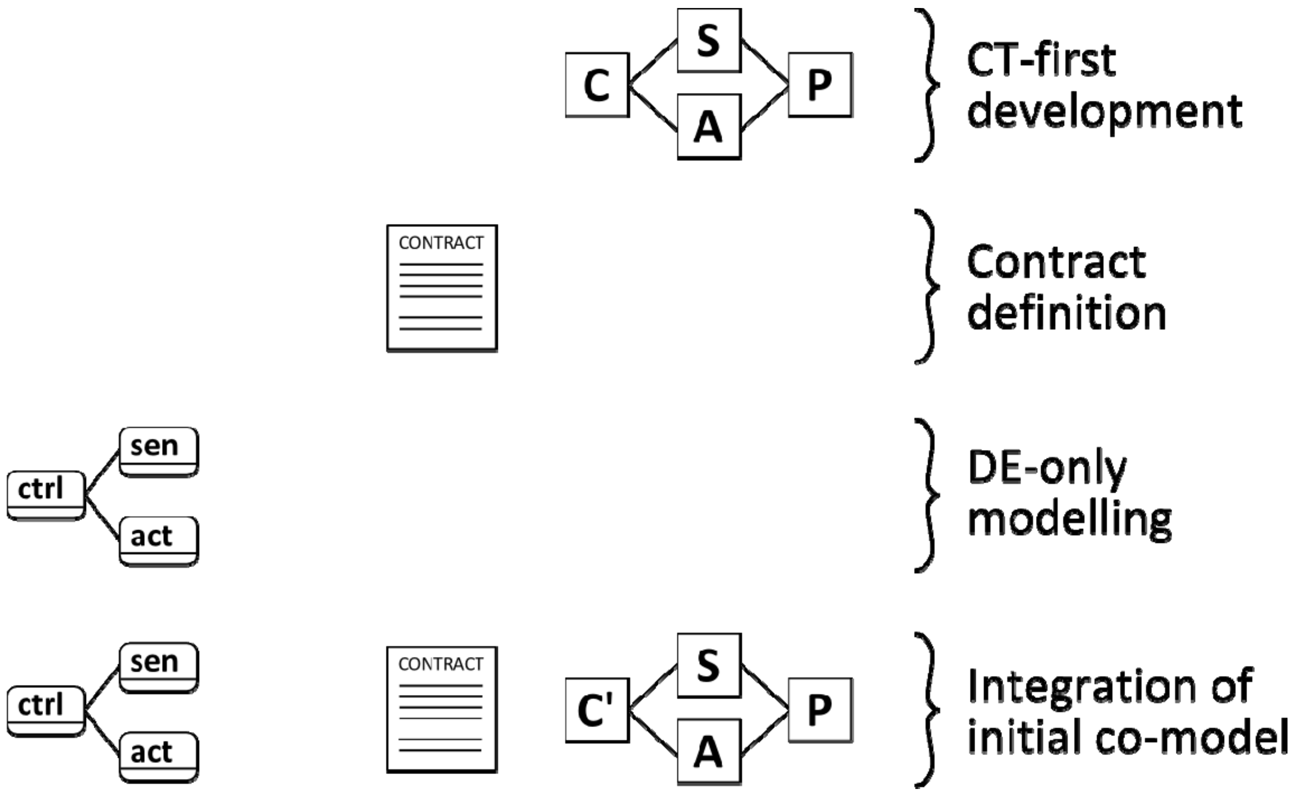
DE-first

- Initial models produced in the discrete-event formalism
- CT model added later



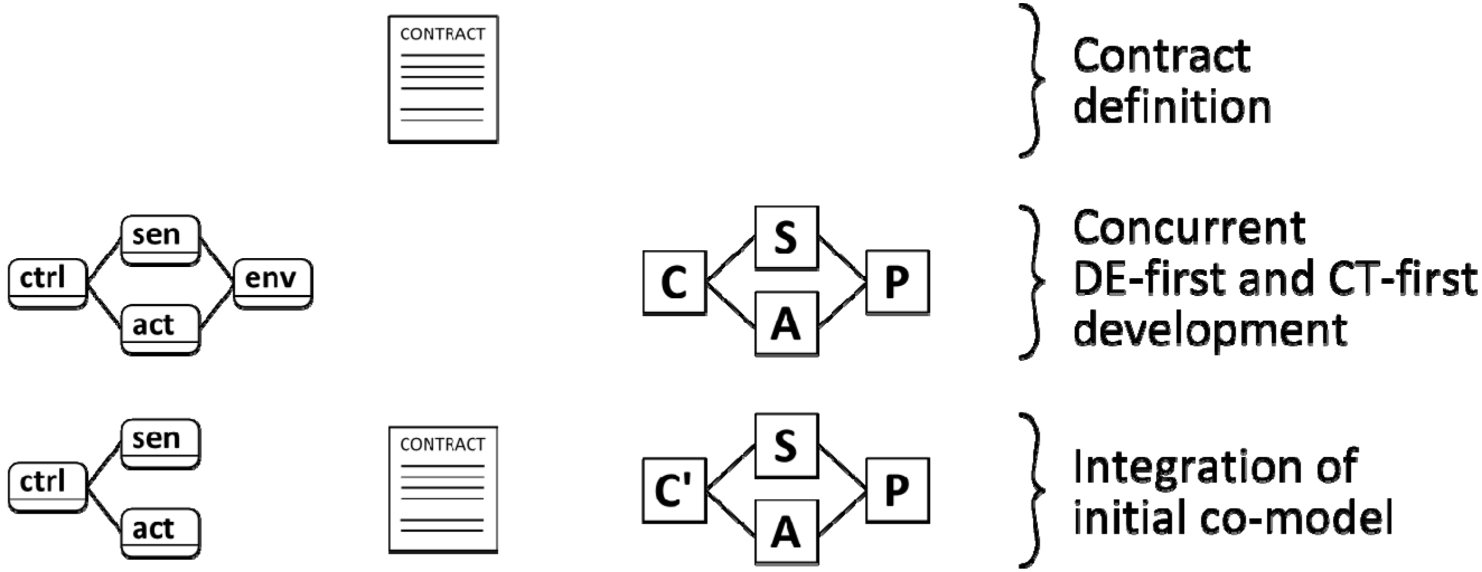
CT-first

- Initial models produced in the CT tool
- DE model introduced later to form a co-model



Contract-first

- Contract defined, acts as a guide
  - allows for early testing of constituent models
- DE- and CT-models developed separately but concurrently
  - following DE-first and CT-first as previously shown
- Constituent models are then integrated into a co-model



Choosing a Path

	Pros	Cons	Use where…
DE-first	complex controller behaviour can be studied early	plant dynamics over-simplified; loop controllers cannot be tuned; rapid increase in environment model complexity	complex DE control needs priority; legacy DE models exist; modeller experience is mainly in DE domain
CT-first	feasibility study; plant dynamics can be studied early on; loop controllers can be tuned	complex DE control cannot be easily studied	feasibility of control unknown; plant dynamics need priority; legacy CT models and/or loop controllers exist; modellers' experience is mainly in CT
Contract-first	a co-model reached early on; constituent models not mutually dependent for testing	contract required early on; extra effort is required in building testing constituent models	integration is required of two legacy models; no legacy models exist; modellers from both domains are available (or have no bias)
Other	a novel approach can fit better with existing practice	limited experience from our existing guidelines	the standard approaches do not fit your development context; legacy models / experience in other formalisms

Summary of Terms (1)

- **model**
  - a more or less abstract representation of a system or component of interest.
- **modelling**
  - the activity of creating models.
- **simulation**
  - symbolic execution of a model.
- **continuous-time simulation**
  - a form of simulation where the state of the system changes continuously through time.
- **discrete-event simulation**
  - a form of simulation where only the points in time at which the state of the system changes are represented.

Summary of Terms (2)

- **co-model**
  - a model comprising two constituent models (a DE sub-model and a CT submodel) and a contract describing the communication between them.
- **contract**
  - a description of the communication between the constituent models of a co-model, given in terms of shared design parameters, shared variables, and common events.
- **co-simulation**
  - the simulation of a co-model.
- **design space exploration (DSE)**
  - the (iterative) process of constructing co-models, performing co-simulations and evaluating the results in order to select co-models for the next iteration.

Summary

- Embedded systems design
  - requires collaborative development
  - analysis of models from different disciplines
  - diverse cultures, abstractions, formalisms
- Crescendo solution is **co-simulation**
  - combining DE models of controllers and CT models of controlled plant
  - allow existing knowledge and skill
  - enable communication between disciplines